Project Writeup

Project title: Advanced lane finding

My project includes the following files:

- vehicle_detection.ipynb containing the script
- writeup_report.pdf summarizing the results
- Video
- Output_images

The goals of this project were:

a) Make a pipeline to detect vehicles in an image and draw bounding boxes around them.
b) Comment on the approach in a report.

Solution approach: I found that the traditional computer vision approaches were very clunky to use and understand. Hence, I used an end to end deep learning model using Faster R-CNN object detection model. Instead of building a detection model from scratch I used the TensforFlow object detection API developed by Google and modified it to suit the project.

Reflection

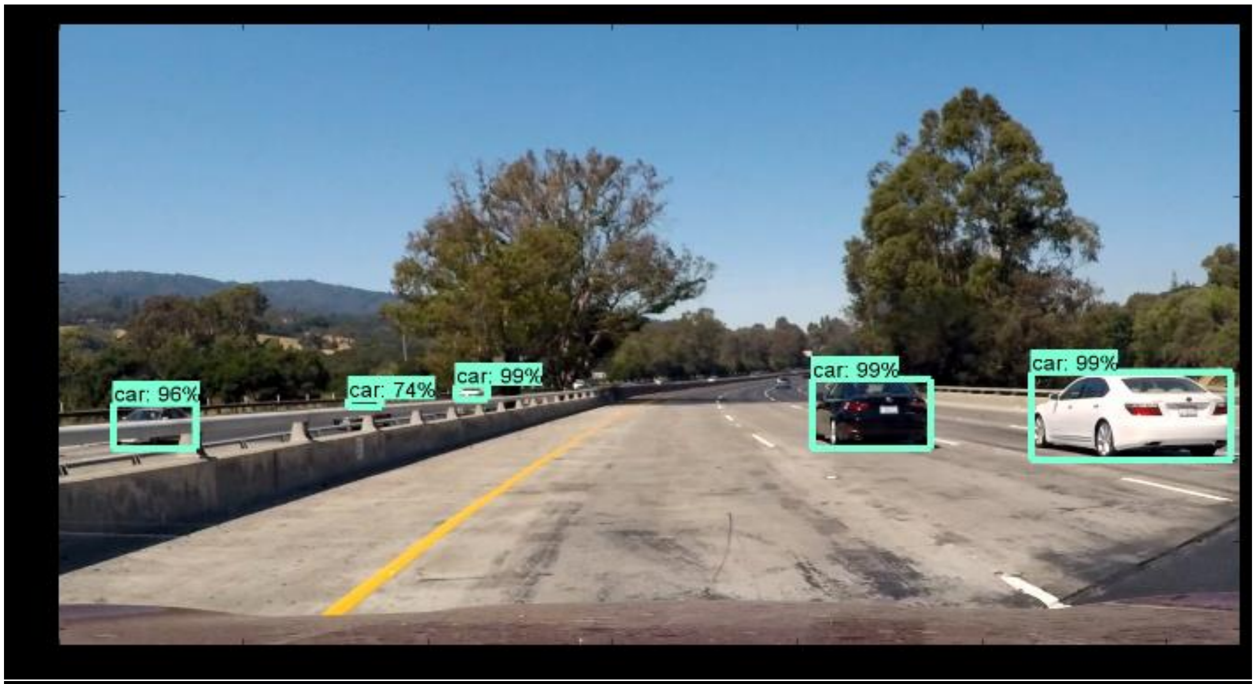
The tensorflow object detection API was used, so I need to first clone the 'github.com/tensorflow/models' repository on my desktop. To make the API work I need to do the following from within the 'object_detection' folder within the repository:

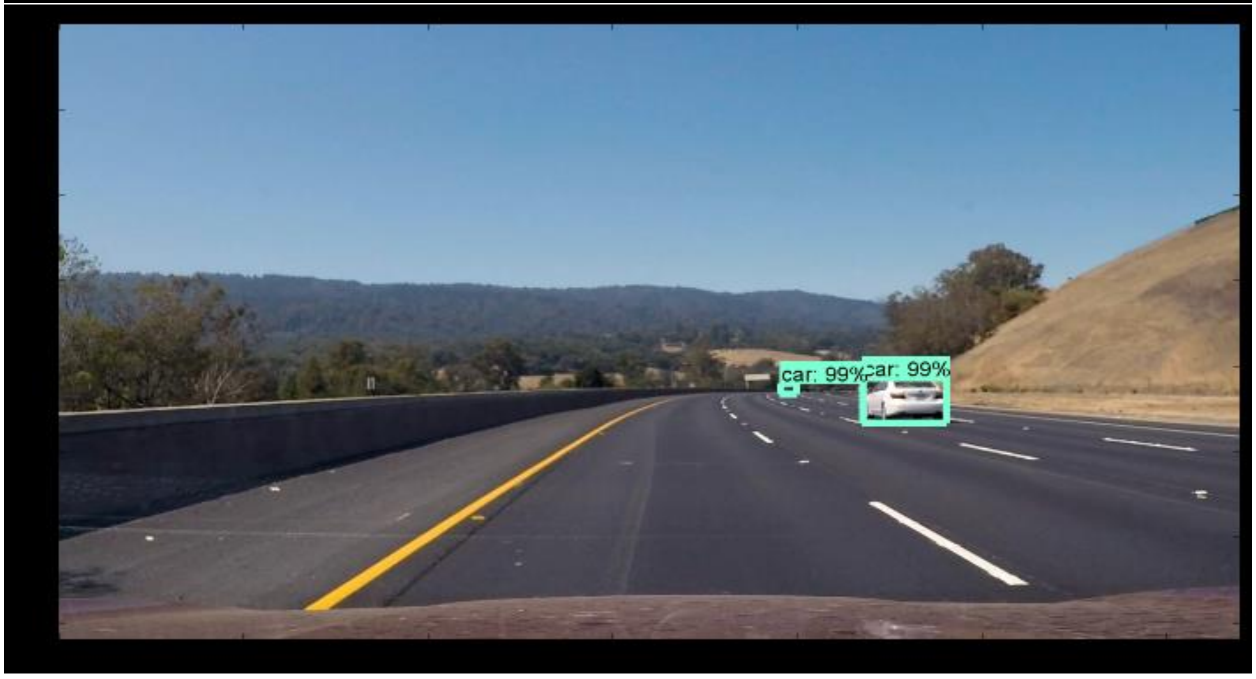
a) Protobuf Compilation
b) Adding the directory containing repository to 'PYTHONPATH' in system variables (on Windows machine).
c) Running the 'setup.py' program from the repository.

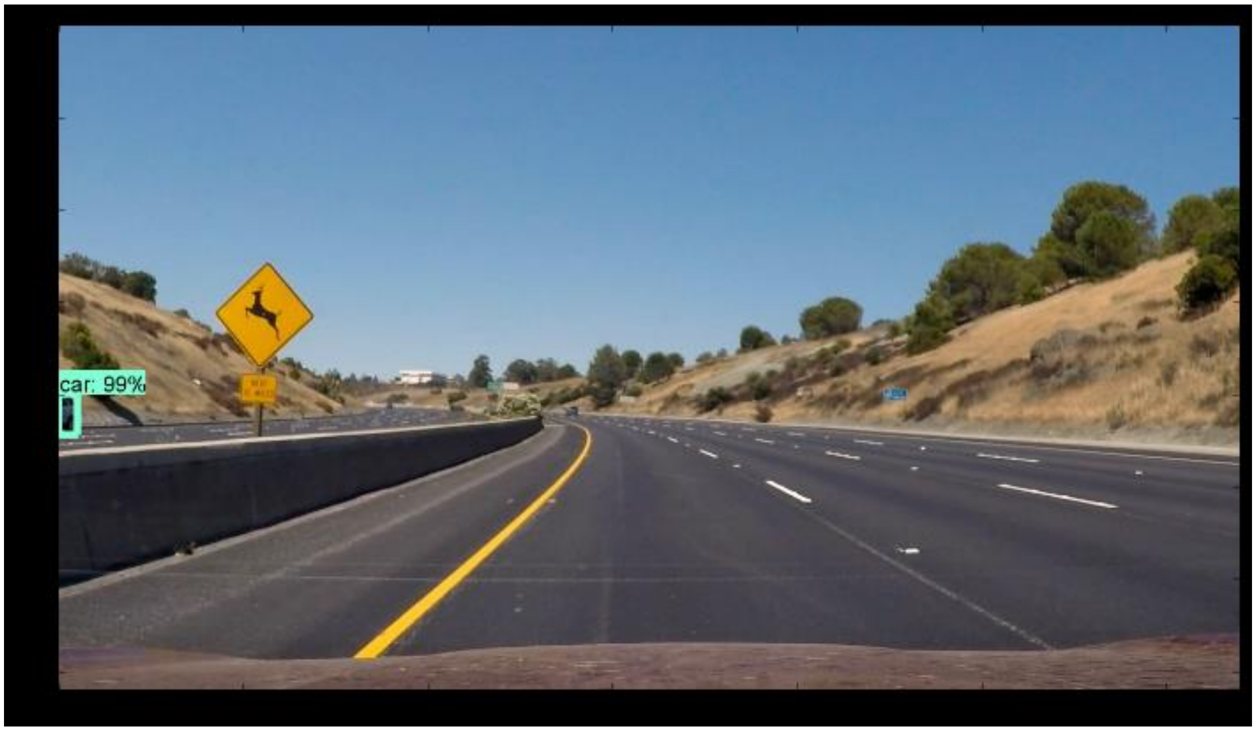
Pipeline description: The software pipeline had the following components.

1. First the necessary dependencies are loaded, and environment is setup. The functions for this step is contained in Cells 1,2 and 3 of the Jupyter notebook.
2. Then the folder name containing the model is specified. The code for this is given in cell 4 of the Jupyter notebook. I used the Faster R-CNN with NASNet model which was downloaded from: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md
3. Then I created a function to import the models into memory as a tensorflow graph called 'detection_graph'. I created anothr function for the operations related to the tensorflow graph. The functions are given in Cell 5 and 6 of the Jupyter notebook.
4. Finally, I created a function that accepts an image and runs the detection model. It also draws boxes around the objects in the image and probabilities associated with the detected objects.

5. The labels assocated with the model were specified in Cell 9 and the detection model function were called in cell 10.
6. For detecting cars in the video I loaded the video using Moviepy and loaded it into the process_image function I had made earlies.
7. Detections that were obtained on the example images are given below. It can be seen that the model is extremely accurate and detects images of cars that were partially obstructed by the divider line.







Identify potential shortcomings with your current pipeline:

The main shortcoming is the time it takes to run the detection model with a GPU. It took 10 hrs for me to process the 50s video clip.

Suggest possible improvements to your pipeline

I can try to use to a smaller model like the mobilenet to improve time but for reduce accuracy.

Conclusion:

The vehicle detection pipeline was implemented using the TensorFlow object detection API using NASNet model. It was found to successfully detect cars with a high degree of confidence.